**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра САПР**

# ОТЧЕТ

**по лабораторной работе №2**

**по дисциплине «Компьютерная графика»**

**Тема: Формирование различных кривых с использованием ортогонального проектирования на плоскость визуализации (экране дисплея)**

Студенты гр. 1302 _____ Марзаева В.И.

_____ Новиков Г.В.

_____ Романова О.В.

Преподаватель _____ Колев Г.Ю.

Санкт-Петербург

2024

**Цель работы**

Сформировать на плоскости кривую Безье на основе задающей ломаной, определяемой 3 и большим количеством точек. Обеспечить редактирование координат точек задающей ломаной с перерисовкой сплайна Безье.

**Теоретическая часть программы**

Кривая Безье – это параметрическое по $t \in [0,1]$ уравнение, являющееся линейной комбинацией базисных функций Безье (представляющих собой базисные полиномы Бернштейна) степени n и контрольных точек $P_i = (x_i, y_i)$:

$$B(t) = \sum_{i=0}^{n-1} P_i * b_{i,n}(t)$$

В связи с трудоемкостью расчетов в современных графических системах обычно используются лишь линейные, квадратичные и кубические кривые Безье, из которых и собираются более сложные кривые.

Поскольку степень кривой всегда равна n (на 1 меньше числа контрольных точек), то увеличение числа контрольных точек приводит к росту степени сплайна, что, в свою очередь, вызывает вычислительные затруднения при расчете биномиальных коэффициентов базисных функций. Основная проблема – это даже не рост вычислительных затрат, а рост значений используемых факториалов. Использование формулы Стирлинга разрешает эту проблему, однако погрешность при ее использовании улучшается до 9-го знака мантиссы лишь к факториалу 12. Соответственно, для повышения устойчивости и был разработан рекурсивный алгоритм де Кастельжо, позволяющий достаточно эффективно рассчитывать полиномы Бернштейна.

Идея рекурсивного алгоритма де Кастельжо построения кривой Безье через параметр t заключается в следующем: каждая точка B(t) кривой находится путем построения сокращающейся последовательности ломаных

$(P_0^{(i)}, P_1^{(i)}, \dots, P_{n-1}^{(i)})$, последняя из которых (выродившись в точку) и даст требуемую точку кривой:

$$(P_0^{(0)}, P_1^{(0)}, \dots, P_{n-2}^{(0)}, P_{n-1}^{(0)}, P_n^{(0)}),$$

$$(P_0^{(1)}, P_1^{(1)}, \dots, P_{n-2}^{(1)}, P_{n-1}^{(1)}),$$

$$(P_0^{(2)}, P_1^{(2)}, \dots, P_{n-2}^{(2)}),$$

$$\dots$$

$$(P_0^{(n-2)}, P_1^{(n-2)}),$$

$$(P_0^{(n-1)}),$$

Где: $P_i^{(0)} = P_i, i = \overline{1,n}$,

$$P_j^{(k)} = (1-t) * P_j^{(k-1)} + t * P_{j+1}^{(k-1)}, j = \overline{1,n-k}, k = \overline{1,n-1}$$

**Пример работы программы**

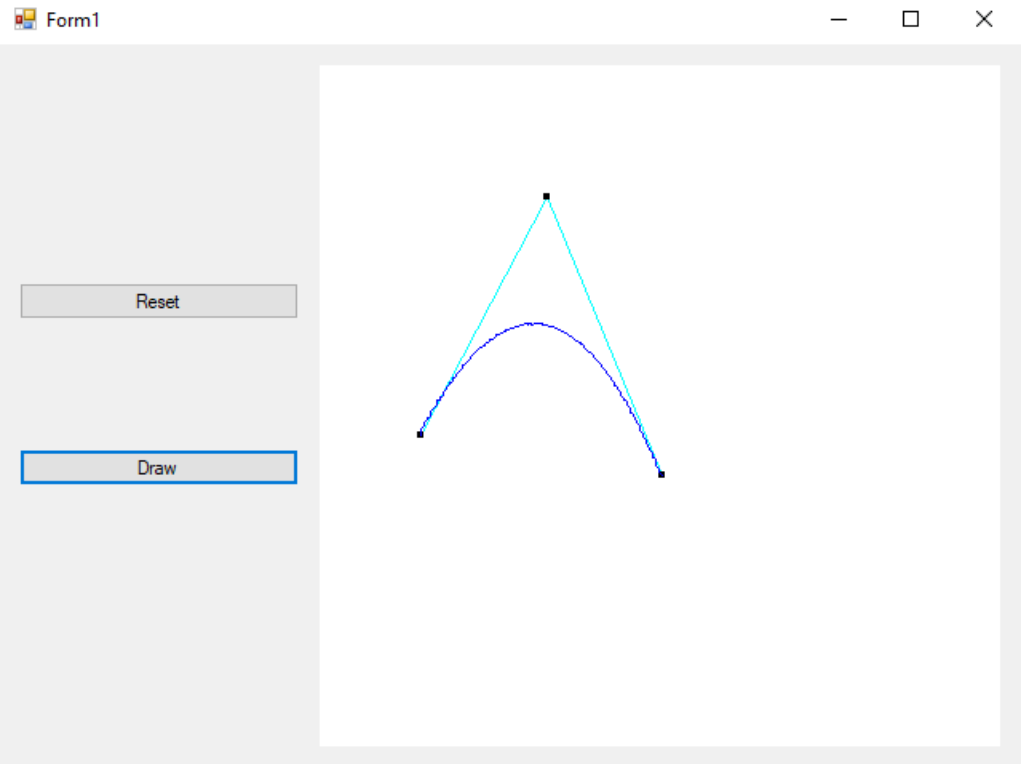


Рис. 1 – Пример работы программы с 3 точками

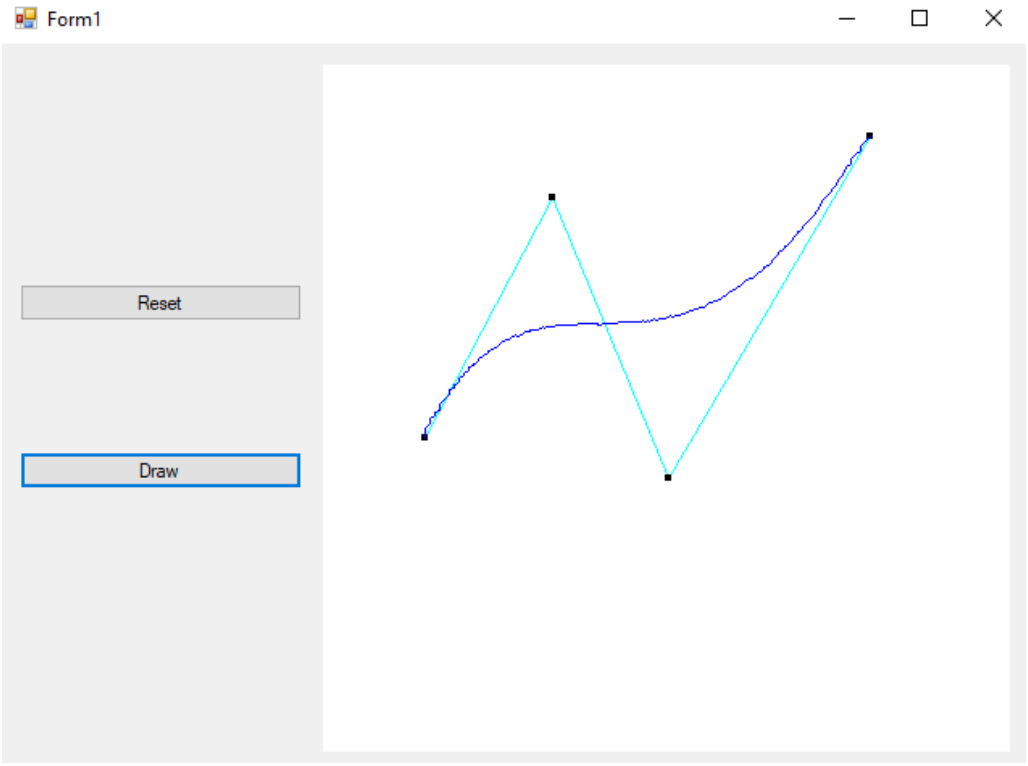


Рис. 2 – Пример работы программы с 4 точками

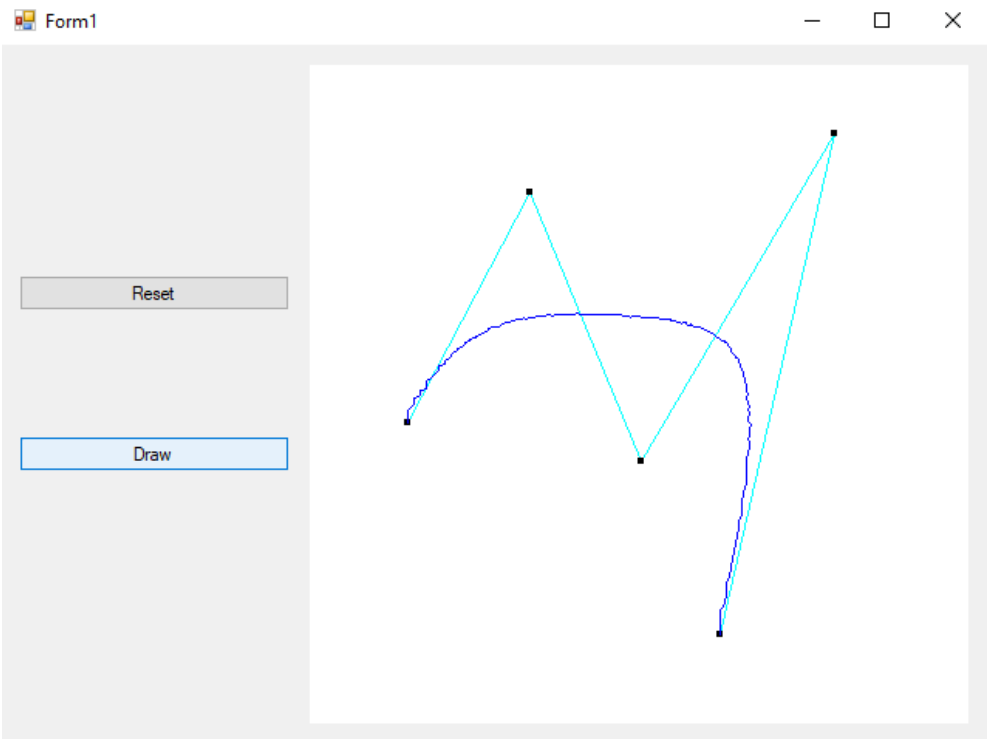


Рис. 3 – Пример работы программы с 5 точками

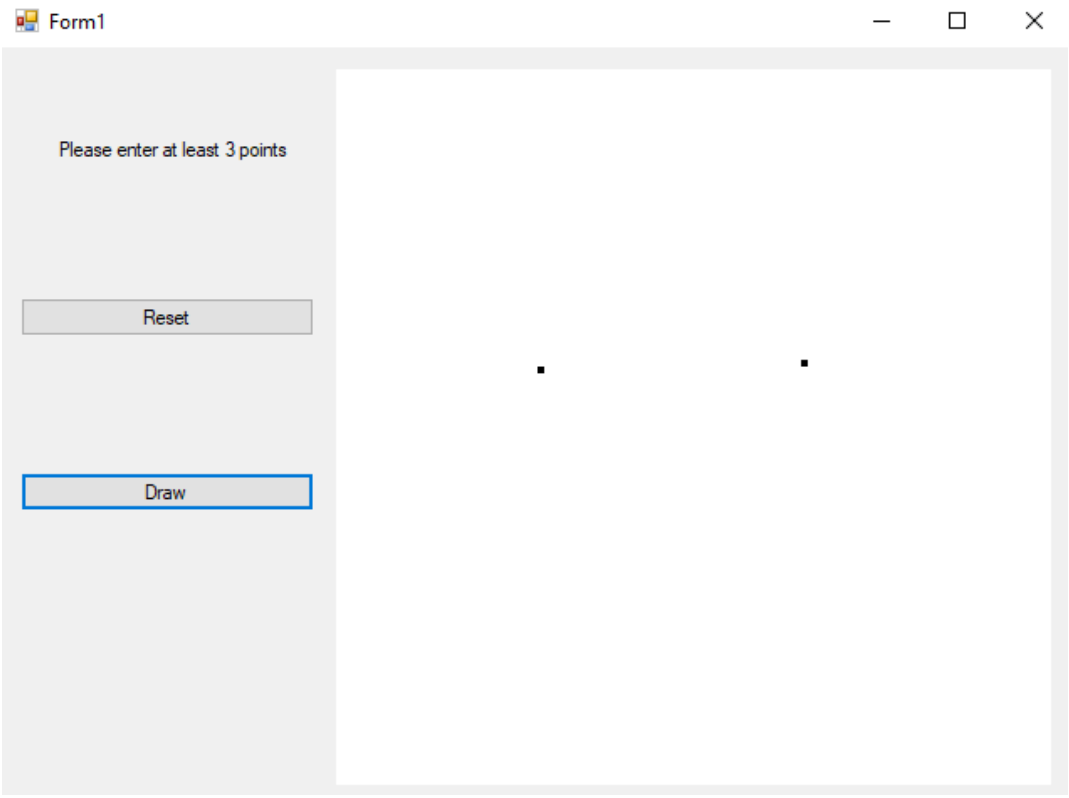


Рис. 4 – Пример работы программы с 2 точками

## Код программы

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace lab2
{
	public partial class Form1 : Form
	{
		Label errorMessageLabel;
		Graphics g;
		SolidBrush brush;
		Pen pen;
		Pen linePen;

		int pointFatness = 4;

		List<Point> points;
		int requiredNumberOfPoints = 3;
		int splineOrder = 2;

		public Form1()
		{
```

```
                        InitializeComponent();
                        errorMessageLabel = label6;
    errorMessageLabel.Text = "";

                        g = pictureBox1.CreateGraphics();
                        brush = new SolidBrush(Color.Black);
                        pen = new Pen(Color.Blue);
                        linePen = new Pen(Color.Cyan);

                        points = new List<Point>();
                }

                private void button1_Click(object sender, EventArgs e)
                {
                        if (EnoughCoordsEntered())
                        {
                                errorMessageLabel.Text = "";
       g.Clear(Color.White);
                                DrawLines();
                                DrawPoints();
                                DrawSpline();
                        } else
                        {
                                errorMessageLabel.Text = "Please enter at least 3 points";
                        }
                }

private void button2_Click(object sender, EventArgs e)
{
    errorMessageLabel.Text = "";
                        points.Clear();
                        g.Clear(Color.White);
}

private void pictureBox1_MouseClick(object sender, MouseEventArgs e)
{
                        points.Add(new Point(e.X, e.Y));
    DrawPoint(e.X, e.Y, pointFatness);
}

                private void DrawPoints()
                {

                        for (int i = 0; i < points.Count; i++)
                        {
                                DrawPoint(points[i].X, points[i].Y, pointFatness);
                        }

                }

                private void DrawLines()
                {
                        for (int i = 0; i < points.Count - 1; i++)
                        {
                                DrawLine(points[i], points[i + 1]);
    }
}

                private void DrawLine(Point p1, Point p2)
                {
    g.DrawLine(linePen, p1, p2);
                }

                private void DrawPoint(int x, int y, int fatness)
                {
    g.FillRectangle(brush, x - fatness / 2, y - fatness / 2, fatness, fatness);
                }

                private void DrawSpline()
                {
```

```
                int numberOfPointsDrawn = 200;
                Point[] resultPoints = new Point[numberOfPointsDrawn];

                float t = 0f;
                float step = 1f / numberOfPointsDrawn;

                for (int i = 0; i < numberOfPointsDrawn; i++)
                {
                        resultPoints[i] = Bezier(points, t);
                        t += step;
                }

                for (int i = 0; i < numberOfPointsDrawn - 1; i++)
                {
                        g.DrawLine(pen, resultPoints[i].X, resultPoints[i].Y, resultPoints[i + 1].X, resultPoints[i +
1].Y);
                }

                int iResult = resultPoints.Length - 1;
                int iPoints = points.Count - 1;
        g.DrawLine(pen, resultPoints[iResult].X, resultPoints[iResult].Y, points[iPoints].X, points[iPoints].Y);
            }

            private Point Bezier(List<Point> P, float t)
            {
                if (P.Count == 1) return P[0];

                List<Point> newP = new List<Point>();
                for (int i = 0; i < P.Count - 1; i++)
                {
          Point point = new Point();
                    point.X = (int)((1 - t) * P[i].X + t * P[i + 1].X);
                    point.Y = (int)((1 - t) * P[i].Y + t * P[i + 1].Y);
                    newP.Add(point);
                }
                return Bezier(newP, t);
            }

            private bool EnoughCoordsEntered()
            {
                return points.Count() >= requiredNumberOfPoints;
            }

    }
}
```

## Выводы

В данной работе с помощью Windows Forms на C# была реализована программа, строящая на плоскости кривую Безье на основе задающей ломаной по трем и более точкам, точки вводятся пользователем при нажатии на экран. При меньшем количестве точек выводится сообщение о слишком малом количестве точек.